Applied Intelligence
https://doi.org/10.1007/s10489-022-04155-2

Buffer-based adaptive fuzzy classifier

Sajal Debnath[1] · Md Manjur Ahmed[1] · Samir brahim Belhaouari[2] · Toshiyuki Amagasa[3] · Mostafijur Rahman[4]

Accepted: 31 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

In the age of a technological revolution, heterogeneous sources are generating streams of data at a high rate, and an online classification of these data can facilitate data mining and analysis. Among the available classifiers, fuzzy-system-based (FSB) classifiers provide remarkable contributions due to their antecedent-consequent rule base structure. The Mamdani and Takagi-Sugeno type structure always uses the identical antecedent portion with fuzzy sets, which are themselves specified by parameterized membership functions driven by logical AND/OR operations. These membership functions are discerned either by experts or from data. However, for online or stream data, using a predefined membership function is not ideal. Meanwhile, a data-cloud has the ability to adopt changes in stream data, which share the same properties as those of a cluster but does not have any predefined shapes or a particular radius; rather, data-cloud offer a more objective representation of real-time data. Moreover, most algorithms with FSB classifiers avoid the presence of temporarily irrelevant data points or data-clouds that can be relevant in the future. In this paper, we develop a novel data-cloud-based classification algorithm for stream data classification called buffer-based adaptive fuzzy classifier (BAFC). The offline training stage of this algorithm can identify data-cloud from a static dataset to construct the AnYa type fuzzy rule. This algorithm is also able to cope with the dynamic nature of stream data. At the online or one-pass training stage, BAFC updates its rule base by creating and merging data-cloud based on its potential area. This algorithm also introduces a recursive formula for calculating data-cloud density with a buffer that is used for storing temporarily irrelevant data clouds. BAFC also uses the online pruning system of data-clouds to address storage problems. This approach can solve the issues associated with the parameterization and redundant rule base for other types of stream data (e.g., sensor data, bank transaction, intruder detection, images and videos, and, stock market and disease prediction) classification algorithms. This two-stage algorithm is evaluated on several benchmark datasets, and the results prove its superiority over different well-established classifiers in terms of classification accuracy (90.82% for 6 datasets and 97.13% for the MNIST dataset), memory efficiency (twice higher than other classifiers), and efficiency in addressing high-dimensional problems.

Keywords Data-cloud · Fuzzy rule · Adaptive classifier · AnYa

1 Introduction

With the revolutionary development of technology, data are being generated through numerous sources, such as computers, smart-phones, autonomous systems, satellites, sensors, social media and other applications [1, 2]. Previous research shows that, more than 2.5 quadrillion megabytes of data were generated daily in 2018 and such amount of data is predicted to grow twice its size every two years according to the International Data Corporation (IDC) [3]. This overwhelming volume of unbounded, structured, or unstructured series of data that arrive continuously are called stream data [1]. However, data mining or classification algorithms encounter challenges in extracting preserved knowledge form stream data three basic features, namely, volume, velocity, and variety [1, 2]. In recent decades, data classification has received much research attention in machine learning [4]. With the introduction of the fuzzy-rule-based (FRB) system, alternative data classification algorithms have emerged, such as k-nearest neighbor (KNN) [5], Bayesian classifiers [6], decision trees [7] , random forest [8], support vector machines [9], and neural-network-based classifiers [10], which are widely used in multiple areas of computational intelligence, including image classification [11], remote sensing [12],

✉ Md Manjur Ahmed
manjur_39@yahoo.com

Extended author information available on the last page of the article.

Published online: 26 October 2022

pattern recognition [13], fault detection [14], user behaviour modeling [15], and handwritten digit and face recognition [16, 17]. However, FRB is preferable for its transparent and interpretive characteristics [18]. Furthermore, online adaptive and evolving fuzzy systems have become prominent techniques for handling or modeling stream data with challenging characteristics [19].

According to their architectures and operating methods, the contemporary FRB techniques can be classified into offline [20, 21] and online [22–25]. Offline classification algorithms can extract knowledge from a static dataset and does not change its learning architecture. As a result, these algorithms are not applicable for manipulating online stream data. Meanwhile, online classification algorithms (incremental and evolving) can be of one pass type [4]. They only store extracted interpretation or information (i.e., rule base [23], key prototype [26], or cloud density [27]) in the system from non-stationary stream data and abandon previously processed samples [22, 23]. To cope with the overwhelming amount of data in real time, online classifiers can address the evolving patterns or trends of data by changing their structures and meta-parameters iteratively [4]. These approaches have been frequently used in different applications for their low memory consumption and computational efficiency. However, the changing pattern of stream data may negatively affect the performance of online classification algorithms.

Many studies have proposed stream data classifiers based on prototype-data and data density. However, most of these classifiers suffer from user-defined parameterization problems, high memory requirements, the "curse of dimensionality" [4, 26, 28, 29], and inability to handle stream data (i.e., by creating irrelevant fuzzy rules). Non-optimal user-defined parameters and irrelevant rule bases can also deteriorate the performance of classifiers. In reality, some data are available in static form, whereas the rest can arrive sequentially as stream data. In this case, learning "from scratch" is inappropriate, and the available static data can be trained offline and assumed knowledge can perform on the stream data [4]. Additionally, an online structure can refute the prior knowledge for non-stationary data by developing their rule base and structure. Thus, an offline (using available static data) and online (using stream data) training-based classification algorithm that adopts meta-parameters recursively and prunes fuzzy rules online with minimum memory and time consumption should be developed. Furthermore, some classifier algorithms may ignore data-clouds that may appear irrelevant but may become relevant in the near future.

This paper proposes the FRB classifier called buffer-based adaptive fuzzy classifier (BAFC) where the algorithm is divided into offline and online training stages. At the offline stage, the 0-order AnYa type fuzzy rule is created by using knowledge extracted from an available static dataset. At the online stage, the rule base identified at the offline stage can be updated continuously by processing stream data. The online procedure belongs to the "one pass" system and supports drifts and/or shits in the data pattern. BAFC does not need any predefined input variable, such as granularity in SOF [4], and reduces its the dependency on the user. A new recursive formula based on cosine dissimilarity is introduced to free the classifier from the "curse of dimensionality." This classifier also handles seemingly irrelevant stream data that may become relevant in the future by using the buffer concept and introduces an efficient and elegant approach for merging existing overlapping data-clouds at the offline and online stages, hence reducing memory and time complexity. This research offers the following contributions:

1. The state-of-the-art classification method SOF [4] uses a predefined input variable (granularity level) as threshold and a greedy approach for ranking the multimodal-density-based list to reduce the efficacy of the classifier. In the proposed system, the Prim's algorithm [30]-based ranking technique and the local average distance of a data-cloud (as threshold) reduce the dependency of BAFC on the user and increase the efficiency of the classifier.
2. A new recursive formula based on unimodal density [31] and cosine dissimilarity is also introduced to facilitate the calculation at the online stage and to free the process from the "curse of dimensionality."
3. At the offline or online stage, some data-clouds (as described in Section 2.2.2) may appear irrelevant but can be relevant in near future, which are not considered by contemporary classifier algorithms. BAFC uses a temporary memory storage for storing these seemingly irrelevant data-clouds. These data-clouds are retrieved from the temporary memory storage if they are found relevant in the future, and by through a pruning operation, the proposed classifier can permanently remove the irrelevant data-clouds. This operation reduces the complexity of creating new data-clouds and minimizes the processing time of stream data.
4. BAFC uses a viable approach to merge data-clouds at the offline stage to form highly compact data-clouds. This classifier also introduces an efficient and elegant technique for merging existing data-clouds at the online stage, which can help reduce the number of overlapping data-clouds and identify the changing patterns of evolving stream data.

Experimental results shows that BAFC outperforms the existing classifiers in terms of accuracy, memory consumption, and processing time and can even generate more compact, transparent, and comprehensible data-clouds. The

rest of this paper is organized as follows. Section 2.1 reviews the literature on stream data and data-cloud-based classification. Section 2.2 presents some existing data classification techniques and theories. Section 3.1 describes the process of deriving the proposed cosine-dissimilarity-based recursive formula, whereas Sections 3.2 and 3.3 explain the methodology of the FRB and data-cloud-based classifiers. Section 4 evaluates and compares the performance of BAFC with that of eight classification methods on eight benchmark datasets. Section 5 concludes the paper.

2 Background study

2.1 Literature review

The literature review focuses on three categories of well-known data classification approaches, namely, (1) FRB, (2) data-cloud-based, and (3) empirical data analysis (EDA)-based approaches.

eClass [32], a prototype/data point-potentiality-based online classifier, is considered a primitive algorithm for classifying stream data that creates fuzzy rules with the most potential data point. In eClass, a fuzzy rule is created via a recursive calculation of prototype potentiality and based on the esteemed value of a Gaussian-bell-type membership function. This classifier has a one-pass architecture and can develop its rule base "from scratch." The limitation and shortcomings of eClass have been improved in [23], where the parameter estimation and pruning operations for a fuzzy rule base are introduced using the global potentiality of a data point. To make the procedure more deterministic, the researchers in [23] used cosine dissimilarity to free eClass from the "curse of dimensionality," whereas [32] used the Euclidean-type distance for calculating data point potentiality. However, these approaches do not solve the problem where membership functions generate some estimated value via fuzzy rule aggregation, where the estimated value significantly differs from the tangible data distribution [27]. The parametric fuzzy classifier for stream data proposed in [33] uses predefined parameters and does not consider high-dimensional data for classification. However, this classifier does not consider seemingly irrelevant data that can become relevant in the near future. Recent studies have used deep learning methods, natural language processing (NLP), and fuzzy contrast set in designing data classifiers [34] to inherit the advantages of both neural networks and fuzzy systems. These classifiers process patient-authored text data by using a fuzzy-rule-based system. In the stock classification [35] and data mining [36] processes, type-2 fuzzy set theory has been used to obtain meaningful information and enhance classification accuracy. A new type-3 fuzzy logic system that uses the interval type-3 fuzzy membership function (MF) was introduced in [37]. Type-3 MFs have more degrees of freedom and higher levels of uncertainty compared with type-2 MFs [38]. However, these classifiers only follow parametric procedures and consider static datasets.

All parameterization problems (i.e., center and spread) of the traditional FRB system have been solved by introducing a data-cloud-based [27] antecedent part in the rule. A data-cloud does not need any membership function or a predefined parameter as the center or spread without any shapes or boundaries similar to clusters. The traditional FRB system can calculate the density of a data-cloud recursively to define the antecedent part, which represents the real data distribution system with a fully data-driven process from stream data. Despite enhancing objectiveness, this classifier leads to less interpretability and loss of information due to the one class, one rule system. A predefined parametric and semi-online classifier [39] uses a data-cloud and the AnYa type fuzzy rule for pain classification but blocks the creation of new rules and data-clouds in the evolving process. [20] introduced an offline-based application of the data-cloud concept and the AnYa type fuzzy rule where no data-cloud merging or deletion is considered. For large amounts of data, setting an inappropriate number of data-clouds will lead to the creation of a low-performing model.

The EDA in [31, 40] introduced essential and alternative data analysis operators that are highly efficient for classification [24] and clustering processes [41]. Data analysis operators, such as cumulative proximity, density, typicality, and standardized eccentricity, are non-parametric. These non-parametric and data-driven operators are extracted from empirical observations of arriving data samples and do not need any previous observations. The clustering process [41] has a higher level of transparency and interpretability where data-clouds, the AnYa type fuzzy rule, and EDA are applied. EDA has been implemented in numerous offline [42] and online [41, 43] applications for anomaly detection, classification, and prediction. By contrast, the online data-cloud merging and pruning in online applications where irrelevant data-clouds and fuzzy rules always deteriorate the performance of models have not yet been considered.

2.2 Related theories

2.2.1 Data analysis and EDA operators

EDA is an evolving method for data analysis [31]. However, EDA operators, such as unimodal and multimodal density, have been successfully applied in different classification and regression approaches [4, 44].

Assume that x is the sample of a real-time dataset or stream X and that $X \in R^n$. At the k^{th} time

instance data space $X^k = \{x_1, x_2, x_3....x_k\}$ and $x_i = [x_{i,1}, x_{i,2}, x_{i,3}....x_{i,n}]$, x is the n-dimensional data sample, and i denotes the time instance when x_i arrives. The unimodal density (D_k) of x_i, $i = [1, k]$ is expressed as [40]

$$D_k(x_i) = \frac{\sum_{j=1}^{k} \pi_k(x_j)}{2K\pi_k(x_i)} = \frac{\sum_{j=1}^{k}\sum_{l=1}^{k} d^2(x_j, x_l)}{2K\sum_{l=1}^{k} d^2(x_i, x_l)} \quad (1)$$

where $D_k(x_i)$ represents the density of data x_i at the k^{th} time instance.

Equation (1) can be defined as the sum of distances from all data samples to all other data samples divided by the sum of distances of a particular data sample to all other existing data samples of the data space at the k^{th} time instance.

In case of static datasets, the data distribution system may contain some data sample X that can be repeated multiple times, that is, $\exists x_i = x_j, i \neq j$. Then, the set of unique data samples will be $\{u\}_{U^c}^c = \{u_1^c, u_2^c, \ldots, u_{U^c}^c\}$, $u_i^c = [u_{i,1}^c, u_{i,2}^c, u_{i,3}^c....u_{i,n}^c]$, $\{u\}_{U_c}^c \subseteq X^c$, where U_c is the number of unique data sample in class c and $\sum_{c=1}^{C} U^c$ is equal to the total unique data samples of the static dataset [4, 44]. The corresponding frequency of a unique dataset will be $\{f\}_{U^c}^c = \{f_1^c, f_2^c, ..., f_{U^c}^c\}$, and the sum of all frequency ($\sum_{i=1}^{U_c} f_i$) will be the sum of data in class c. Therefore, the *Multimodal density* (D_c^{MM}) of the unique data sample u_i can be defined as [40]

$$D_c^{MM}(u_i^c) = \frac{f_i^c \sum_{j=1}^{X^c}\sum_{l=1}^{X^c} d^2(x_j^c, x_l^c)}{2X^c \sum_{l=1}^{X^c} d^2(u_i^c, x_l^c)} \quad (2)$$

where i = 1,2,3...U^c.

This study uses *Multimodal density* [40] to calculate the density of a data-cloud from a static dataset and *Unimodal density* for online or real-time situations.

Section 3.1 introduces a new recursive formula for online density calculation using cosine dissimilarity.

2.2.2 Data-cloud and AnYa type fuzzy rule

The data-cloud introduced in [27] has demonstrated success in numerous applications [24]. A data-cloud can identified through some set of data with common properties of a cluster but does not have any predefined boundaries or a particular shape. However, in this study, a data-cloud is presented as either an ellipses or circle. Compared with traditional membership functions, data-cloud have a much more objective representation of real data distribution. By calculating the unimodal density (described in Section 3.1) of stream data x_k, one can identify to which data-cloud this stream data will belong.

Since, the proposed algorithm is recursive (recursively updates the meta-parameters of a data-cloud), BAFC does not store past data at the online training stage. Instead, some information extracted from the data through a variable is needed for the i^{th} data-cloud of class c, including the density of center (D_i^k), data-cloud center (dcc_i^k), number of samples (S_i^k), average distance between center and each sample (Ad_i^k), and age of a data-cloud (Age_i^k) at the k^{th} time instant.

An alternative approach for the widely used FRB system is the AnYa type FRB system introduced in [27]. AnYa type fuzzy rules are highly objective, non-parametric, compact, and does not need any predefined membership function. The AnYa type fuzzy rule can be expressed as [27]

$$\textbf{IF } (x \sim dcc_1^c)\textbf{OR}(x \sim dcc_2^c)\textbf{OR}...\textbf{OR}(x \sim dcc_N^c)\textbf{THEN } Class_c \leftarrow x \quad (3)$$

where dcc_i^c denotes the i^{th} data-cloud of the c^{th} class for an n-dimensional input data vector, x =[x_i, x_2, x_3...x_n], N is the number of data-clouds extracted from data samples that may vary from one class to another, and '$\sim$' denotes the similarity between x and the i^{th} data-cloud.

3 Buffer-based adaptive fuzzy classifier

This section discusses in detail the recursive formula for density calculation, the offline and online training procedures, and the decision-making procedure of the proposed BAFC. A broad description of the complexity analysis of the proposed BAFC algorithm is also presented at the end of this section. The offline training will be processed with a static dataset, and the variables will be prepared per class for use at the online training stage. All calculations at the online training stage will be conducted at the k^{th} time instance.

3.1 Theories

This study introduces a cosine-distance-based formula to facilitate an efficient online calculation and to free the proposed BAFC from the "curse of dimensionality." Assume that the distance type is cosine distance, and the recursive calculation of *Unimodal density* (D_k) is expressed as [40]

$$D_k(x_i) = \frac{\sum_{j=1}^{k}\sum_{l=1}^{k} d^2(x_j, x_l)}{2K\sum_{l=1}^{k} d^2(x_i, x_l)}, i = 1, 2, 3...k; \quad (4)$$

$$= \frac{\sum_{j=1}^{k}\sum_{l=1}^{k} d^2(x_j, x_l)}{2K\sum_{l=1}^{k}\left(1 - \frac{\sum_{j=1}^{n} x_{ij}x_{lj}}{\sqrt{\sum_{j=1}^{n} x_{ij}^2 \sum_{j=1}^{n} x_{lj}^2}}\right)^2}$$

Considering and simplifying the denominator (Dnr) part yield

$$\begin{aligned} Dnr &= 2K\sum_{l=1}^{k}\left(1-\frac{f_l}{hg_l}\right)^2 \\ &= 2K\sum_{l=1}^{k}\left(1-\frac{2f_l}{hg_l}+\left[\frac{f_l^2}{h^2g_l^2}\right]\right) \\ &= 2K\left(k-\frac{1}{h}\left(\sum_{l=1}^{k}\frac{2f_l}{g_l}+\frac{1}{h}\sum_{l=1}^{k}\left[\frac{f_l^2}{g_l^2}\right]\right)\right) \\ &= 2K\left(k-\frac{1}{h}\left(2B_k+\frac{1}{h}D_k\right)\right) \end{aligned} \tag{5}$$

where $f_l = \sum_{j=1}^{n} x_{ij}x_{lj}$, $h = \sqrt{\sum_{j=1}^{n} x_{ij}^2}$, $g_l = \sqrt{\sum_{j=1}^{n} x_{lj}^2}$, $B_k = \sum_{l=1}^{k}\frac{f_l}{g_l}$, and $D_k = \sum_{l=1}^{k}\frac{f_l^2}{g_l^2}$. By observing the variables f_l and g_l, assume that they depend on all the data samples represented by l, but variable h only depends on the arrival of the x_i data attribute. In addition, similar to [26], both B_k and D_k can be calculated recursively as

$$B_k = \sum_{j=1}^{n} x_{ij}b_{kj};\ b_{kj} = b_{(k-1)j} + \sqrt{\frac{x_{kj}}{\sum_{l=1}^{n}(x_{kl})^2}};\ b_{1j} = \sqrt{\frac{x_{kj}}{\sum_{l=1}^{n}(x_{kl})^2}}; \tag{6}$$

and

$$D_k = \sum_{j=1}^{n} x_{kj}\sum_{p=1}^{n} x_{kp}d_{kj}^{p};\ d_{kj}^{p} = d_{(k-1)j}^{p} + \frac{x_{kj}x_{kp}}{\sum_{l=1}^{n}(x_k^l)^2};\ d_{1j}^{1} = \frac{x_{1j}x_{11}}{\sum_{l=1}^{n}(x_1^l)^2} \tag{7}$$

The recursive function of $Unimodal\ density$ is then formulated as

$$\begin{aligned} D_k(x_i) &= \frac{\sum_{j=1}^{k}\sum_{l=1}^{k} d^2(x_j, x_l)}{2K\left(k-\frac{1}{h}\left(2B_k+\frac{1}{h}D_k\right)\right)} \\ &= \frac{\sigma_{k-1}+2\gamma_k}{2K\left(k-\frac{1}{h}\left(2B_k+\frac{1}{h}D_k\right)\right)} \end{aligned} \tag{8}$$

where the distance between the k^{th} data to all other data in the data space is $\gamma_k = k - \frac{1}{h}\left(2B_k + \frac{1}{h}D_k\right)$. The distance among all pairs for the stream data is $\sigma_k = \sigma_{k-1} + 2\gamma_k$, and the σ_1 for the online training stage is computed as the sum of distance of all pairs of available static data at the offline stage.

In addition, for the online process, we have to calculate the density of existing data clouds. We can use the variable and density calculated at the $(k-1)^{th}$ time instance as (where dcc refers the data-cloud center, $dcc \in X$)

$$\begin{aligned} D_{k-1}(dcc^*) &= \frac{\sum_{j=1}^{k-1}\sum_{l=1}^{k-1} d^2(x_j, x_l)}{2(K-1)\sum_{l=1}^{k-1} d^2(dcc^*, x_l)} \\ D_{k-1}(dcc^*) &= \frac{\sigma_{k-1}}{2(K-1)\sum_{l=1}^{k-1} d^2(dcc^*, x_l)} \\ \sum_{l=1}^{k-1} d^2(dcc^*, x_l) &= \frac{\sigma_{k-1}}{2(K-1)D_{k-1}(dcc^*)} \end{aligned} \tag{9}$$

Using (9), the density of the existing data-cloud at the k^{th} time instance can be calculated as

$$\begin{aligned} D_k(dcc^*) &= \frac{\sum_{j=1}^{k}\sum_{l=1}^{k} d^2(x_j, x_l)}{2K\sum_{l=1}^{k} d^2(dcc^*, x_l)} \\ &= \frac{\sigma_{k-1}+2\gamma_k}{2K\left[\sum_{l=1}^{k-1} d^2(dcc^*, x_l) + d^2(dcc^*, x_k)\right]} \\ &= \frac{\sigma_k}{2K\left[\frac{\sigma_{k-1}}{2(K-1)D_{k-1}(dcc^*)} + d^2(dcc^*, x_k)\right]} \end{aligned} \tag{10}$$

where $d^2(dcc^*, x_k)$ is the square of the distance between the newly arrived data x_k and all existing data-cloud centers dcc^*, and $D_{k-1}(dcc^*)$ is the density of the data-cloud at the $k-1^{th}$ time instance. Using (10), the online training stage can update the existing data-cloud center, which does not need any previous data.

3.2 Offline training

This proposed BAFC constructs a potential data-cloud from each class independently and creates an AnYa type fuzzy rule (as (3)) based on the data-cloud center. The data-clouds identified from each class do not affect one another. As mentioned earlier in Section 2.2.1, all EDA operators are operated in the k^{th} time instance for the c^{th} class of data at the online training stage, but at the offline training stage, a static dataset is considered.

First, the proposed classifier determines a superior data sample (unique data that satisfy condition 2) from the dataset using the multimodal density [40] per class. A ranking system is then built by using Prim's algorithm [30], where the maximum density ($\arg\max_{i=1}^{U^c}(D_c^{MM}(u_i^c))$) of a unique data is the root node of the tree. Figure 1 illustrates a tree generated by using 100 class 1 data from the letter recognition dataset. The node number denotes the serial number of unique data in the static dataset, and node-65 is

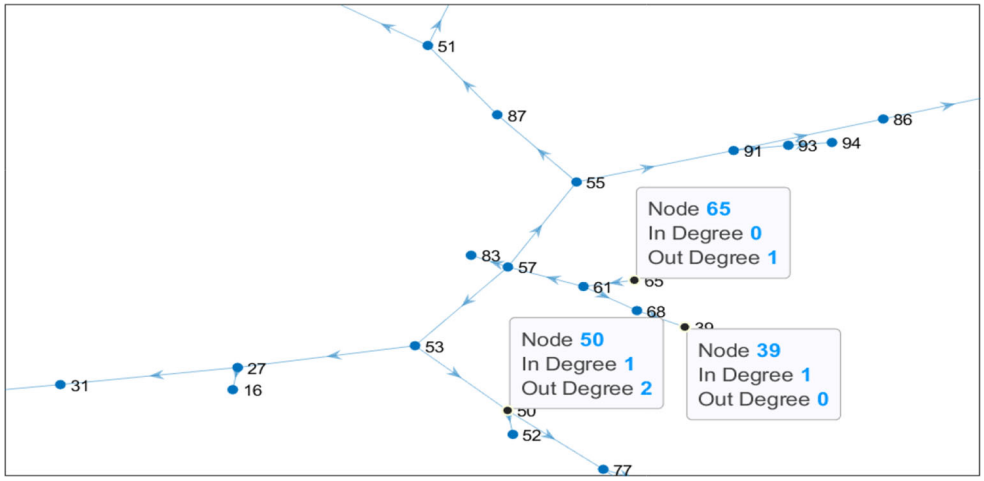


Fig. 1 Tree generated from 100 class 2 data from letter recognition [45] dataset

the root. The sign "←" indicates the child node of a parent node, and the parent node can have more than one child node.

Condition 1

$$\textbf{IF } (D_c^{MM}(node_i) > D_c^{MM}(node_{i+1})) \textbf{ AND}(D_c^{MM}(node_i) > D_c^{MM}(node_{i-1})) \textbf{ THEN} node_i \in dcc_c \quad (11)$$

By applying condition-1 [4] and depth-first search on the tree (similar to Fig. 1), the identified unique data points are assigned in dcc^c for the c^{th} class as a data-cloud center. Now, data-clouds are constructed by using all the sample data and by applying the widely used Voronoi tessellation [46] as

$$\textbf{Cloud } = \arg\min_{i=1}^{m}(d(x_j, dcc_i^c)); x_j \in X^c \quad (12)$$

The cloud center always grabs its nearby data samples to make an effective data-cloud, and the average distance between a cloud center and its data-sample is

$$Ad_i^c = \frac{\sum d(dcc_i^c, x_j)}{\text{number of data sample in } i^{th} \text{ data-cloud}} \quad (13)$$

If any cloud exits into the potential area (i.e., the distance between two cloud centers is less than or equal to Ad_i^c of any of those two data-clouds) of another cloud, then according to condition-2 (14), those clouds will be merged into one i^{th} data-cloud. The merging of data-clouds at the offline training stage prevents the system from growing data clouds uncontrollably.

Condition 2

$$\textbf{IF } \exists\, (d(dcc_i^c, dcc_{j*}^c) \textbf{ AND } i \neq j) \leq Ad_i^c \textbf{ THEN } \forall (dcc_{j*}^c) \in dcc_i^c \quad (14)$$

If any data-cloud satisfies Condition-2, then the meta-parameters for the i^{th} data-cloud will be updated as

$$S_i^c = S_i^c + S_{j*}^c; dcc_c^i = \frac{\sum X_i^c}{S_i^c}; Ad_i^c = \frac{\sum d(dcc_i^c, X_i^c)}{S_i^c} \quad (15)$$

where S_i^c denotes the number of data samples in cloud i of class c, and S_{j*}^c denotes the total number of data samples of all j^{th} cloud in class c that satisfy condition-2. In sum, condition-2 indicates that one or many j^{th} data-clouds can be merged when the center of the j^{th} data-cloud is situated inside the potential area of the i^{th} data-cloud. After updating the meta-parameters (e.g. σ^c, γ^c and $D(dcc_i^c)$) of all classes, the offline training stage prepares the AnYa type fuzzy rule base as shown in (3).

Algorithm-1 presents the overall offline training architecture.

3.3 Online training

The architecture of the evolving training and the removal of irrelevant data-clouds are explained in detail as follows. As this online approach is "one-pass," it will not store previous data samples ; instead, this approach updates the data-cloud and meta-parameters obtained from the offline training stage. The irrelevant data-clouds will be stored in temporary memory storage or buffer and, within some conditions, will be retrieved or removed permanently. Data-clouds are also merged when several conditions are satisfied to enhance the computational and memory efficiency of this approach. Before the evolving training process, age of data-cloud (Age_0^c) will be initialized as zero for each class. Then, a data-cloud will be considered as either relevant or irrelevant based on its age. Moreover, in the following process, the

Algorithm 1 Offline Training.

Input: Static dataset.
Output: Data-cloud, Average distance of each data-cloud (Ad_i^c), data-cloud center (dcc_i^c), support of each data-coud (S_i^c), σ^c and γ^c Find out unique data samples.
Calculate multimodal density ($D_c^{MM}(u_i^c)$) for each unique data sample .
Prepare tree to rank the unique data sample using Prim's algorithm.
Apply condition-1 (11) in the tree to find out superior unique data sample.
Prepare data-cloud using (12).
for *all the existing data-cloud apply condition-2 (14)* **do**
  if *condition-2 is satisfied* **then**
    Update i^{th} data-cloud parameters (Ad_i^c, S_i^c and dcc_i^c) using (15).
    Delete all j^{th} data-cloud.
  end
end
Create AnYa type data-cloud-based fuzzy rule using (3).
Calculate per class parameter σ^c and γ^c.

online training process is assumed to be executed on a sample data belonging to the c^{th} class.

For each arriving data sample, the system calculates its unimodal density using the recursive (8) with variables σ_k^c and γ_k^c. At the same time, the density of the existing data-cloud should be updated recursively using the density of the $(k-1)^{th}$ time instance using (10).

Condition 3 [4]:

$$\textbf{IF } (D_k^c(x_k^c)) > \max(D_k^c(dcc^*)) \textbf{ OR } (D_k^c(x_k^c)) < \min(D_k^c(dcc^*)) \textbf{ OR } (d(x_k^c, dcc^*) > \max(Ad_k^c)) \textbf{ THEN } x_k^c \in dcc^c \quad (16)$$

Condition-3 is satisfied when the newly arrived k^{th} data contain either maximum or minimum density, and when the distance between the k^{th} data x_k^c and with any existing data-cloud is greater than maximum average distance of all data-clouds, a new data-cloud will be created using the aforementioned variables as follows:

$$dcc_{new}^c = x_k^c;\ Ad_{new}^c = 0;\ Age_{new}^c = 0;\ S_{new}^c = 1;\ D_{new}^c = D_k^c(x_k^c) \quad (17)$$

where dcc_{new} is the center of newly created data-clouds, Ad_{new} is the average distance (initially zero) between the center to its data sample, Age_{new} is the age of data-clouds, and D_{new} is the density calculated by the recursive (8).

Condition-3 will not be satisfied when the newly arrived data sample is nearest of a existing data-cloud and nearest data-cloud will be identified as, $i = \arg\min\,(d(x_k^c, dcc_{*,k}^c))$. The parameters or variables of the identified data-cloud are updated at the k^{th} time instance as

$$\begin{aligned} dcc_{i,k}^c &= \frac{(dcc_{i,k-1}^c * S_{i,k-1}^c) + x_k^c}{S_i^c + 1}; \\ Ad_{i,k}^c &= \frac{(Ad_{i,k-1}^c * S_{i,k-1}^c) + d(x_k^c, dcc_{i,k-1}^c)}{S_{i,k-1}^c + 1}; \\ sk_{i,k}^c &= sk_{i,k-1}^c + k;\ S_{i,k}^c = S_{i,k-1}^c + 1; \end{aligned} \quad (18)$$

where sk denotes the sum of time index when the k^{th} data arrive in a specific data-cloud and is needed for recursively calculating the age of a data-cloud.

The age [0,k] of a data-cloud is then calculated recursively as follows to determine whether this data-cloud is relevant or not:

$$Age_{i,k}^c = k - \frac{sk_{i,k}^c}{S_{i,k}^c} \quad (19)$$

Similar to [23], if the data sample is ascribed to an existing data-cloud, then its age will be smaller, and if no data sample is ascribed to an existing data-cloud, its age will increase by 1. By following the age of a data-cloud, one can easily identify whether this data-cloud is relevant or not, and the age never exceeds k. The following condition-4 determines which data-cloud will be stored in the temporary memory storage (buffer) as a seemingly irrelevant data-cloud.

Condition 4

$$\textbf{IF } (Age_{i,k}^c) > \overline{Age_k^c} + std(Age_k^c) \textbf{ THEN } dcc_{i,k}^c \in tmp\_str \textbf{ AND } s\_age_{i,k}^c = 0.5 \quad (20)$$

where i is the data-cloud of a specific class c, $\overline{Age^c_{i,k}}$ denotes the mean age of the i^{th} data-cloud, and $std(Age^c_{i,k})$ is the standard deviation of the age of the i^{th} data-cloud. tmp_str refers to the temporary storage or buffer, and s_age refers to the survival_age of a data-cloud in the buffer. According to the above condition, a data-cloud is considered irrelevant when its age exceeds the "one-sigma" value and will be stored in buffer with a survival_age of 0.5. When the survival_age is less than or equal to 0, the data-cloud will be permanently removed from the system. As mentioned earlier, the proposed system retrieves a data-cloud from the buffer and removes its survival_age when the following condition is satisfied:

Condition 5

$$\textbf{IF } (Age^c_{i,k}) < \overline{Age^c_k} + std(Age^c_k) \textbf{ THEN } Retrieve \leftarrow dcc^c_{i,k} \textbf{ AND } Remove \leftarrow s\_age^c_{i,k} \quad (21)$$

A data-cloud is retrieved from the buffer when it is confirmed to be relevant to the current stream data.

For every iteration, the survival_age of a data-cloud stored in the temporary memory storage is reduced by $\frac{1}{\text{decay}}$ [47]. This process can identify a dying data-cloud that is completely irrelevant to the current data flow or stream for a long period. A dead data-cloud is identified when its survival_age ($s\_age^c_{i,k}$) is less then or equal to zero and will be removed permanently from the system. The process of identifying and removing dead data-clouds can be formulated as

Condition 6

$$\textbf{IF } s\_age^c_{i,k} \leq 0 \textbf{ THEN } Remove \leftarrow dcc^c_{i,k} \quad (22)$$

From condition-6, the irrelevant data-clouds with a non-positive survival_age are identified and completely removed from the buffer.

The proposed system can merge or combine two extremely nearest (overlapping) data-clouds to reduce the number of data-clouds while simultaneously extracting the dynamic or changing the properties of data. A data-cloud merging operation is executed when the distance between two nearest data-clouds is significantly low. The condition for this fully autonomous process is

Condition 7

$$\textbf{IF } d(dcc^c_{i,k}, dcc^c_{j,k}) < \frac{Ad^c_{i,k}}{2} + \frac{Ad^c_{j,k}}{2} \textbf{ THEN } dcc^c_{j,k} \in dcc^c_{i,k}; \quad j = \arg\min_{j=1}^{N^c} d(dcc^c_{i,k}, dcc^c_{j*,k}); \quad (23)$$

The above condition shows that the i^{th} and j^{th} data-clouds are only merged when the distance between them is less than the sum of their half of local average distance, and the j^{th} data-cloud is selected based on its minimum distance from the i^{th} data-cloud. Figure 2 illustrates the procedure of merging data-clouds online. In Fig. 2a, both data-clouds satisfied condition-7, and the merged data-cloud is shown in Fig. 2b.

This merging process prevents an uncontrolled growth of data-clouds, hence making the system more dynamic. When two data-clouds are merged, their variables should be updated as

$$\begin{aligned} dcc^c_{i,k} &= \frac{(dcc^c_{i,k} * S^c_{i,k}) + (dcc^c_{j,k} * S^c_{j,k})}{S^c_{i,k} + S^c_{j,k}}; \\ Ad^c_{i,k} &= \frac{Ad^c_{i,k} + Ad^c_{j,k} + d(dcc^c_{i,k}, dcc^c_{j,k})}{2}; \\ Age^c_{i,k} &= k - \frac{(sk^c_{i,k} * S^c_{i,k}) + (sk^c_{j,k} * S^c_{j,k})}{S^c_{i,k} + S^c_{j,k}}; \\ sk^c_{i,k} &= sk^c_{i,k} + sk^c_{j,k}; \; S^c_{i,k} = S^c_{i,k} + S^c_{j,k}; \end{aligned} \quad (24)$$

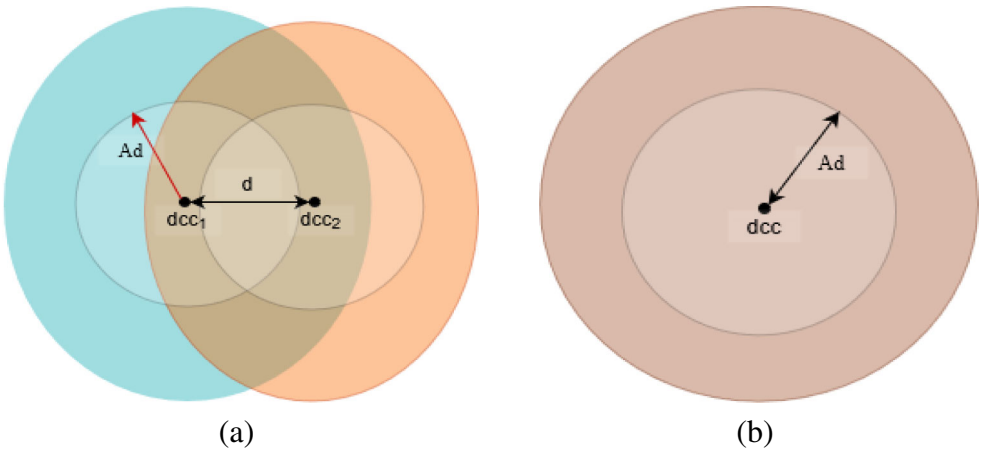


Fig. 2 Merging of data-clouds in online mode, where (a) presents the state before the merging, and (b) describes the state after merging data-clouds 1 and 2 according to condition 7 (23)

Algorithm 2 Evolving Training.

Input: Data sample x_k^c as stream data and parameters (Ad_{cloud}^c, dcc_{cloud}^c, S_{cloud}^c, σ^c and γ^c) from offline training stage.
Output: AnYa type data-cloud based fuzzy rule.
while *Data stream not ends* **do**
  Initialize Age_{ik}^c, sk_{ik}^c, B_k^c and D_k^c .
  Calculate unimodal density of arrived data $D_k^c(x_k^c)$ applying Eq. (8).
  Calculate density of existing data-clouds $D_k^c(dcc^*)$ using recursive Eq. (10).
  if *condition-3 (16) is satisfied* **then**
    Create new data-cloud with meta-parameters using Eq. (17).
  else
    Insert in a existing data-cloud and update meta-parameters of Eq. (18).
  end
  for *Every existing data-cloud* **do**
    Update age (Age_{ik}^c) recursively using Eq. (19).
  end
  for *Every existing data-cloud* **do**
    if *condition-4 (20) is satisfied* **then**
      Move data-cloud into Buffer.
      Assign survival age of the data-cloud, $s\_age_{i,k}^c = 0.5$.
    end
  end
  for *Every data-cloud in tmp_str (Buffer)* **do**
    Reduce decay from survival age as $s\_age_{i,k}^c = s\_age_{i,k}^c$ - $\frac{1}{\text{decay}}$.
    if *condition-5 (21) is satisfied* **then**
      Retrieve data-cloud from tmp_str (buffer).
      Remove survival age of the data-cloud.
    end
  end
  for *Every data-cloud in tmp_str (Buffer)* **do**
    if *condition-6 (22) is satisfied* **then**
      Remove data-cloud permanently.
    end
  end
  for *Every existing data-cloud* **do**
    if *condition-7 (23) is satisfied* **then**
      Merge data-clouds i^{th} and j^{th} into i^{th} data cloud.
      Update properties of i^{th} data-cloud with Eq. (24).
    end
  end
  Create AnYa type data-cloud based fuzzy rule using Eq. (3).
end

After the online merging operation of a data-cloud, its AnYa type fuzzy rule base is updated according to (3), and either the next data sample is selected or the current data-cloud will be shifted at validation stage.

Algorithm-2 presents the online training architecture.

3.4 Decision making procedure

For the arriving data sample (x), each active fuzzy rule provides a firing strength by computing the similarity between x and data-clouds denoted by $\gamma^c(x)$ (c = 1, 2,

3...C), which is determined by [4]

$$\gamma^c(x) = \max_{dcc \in \{dcc^C\}} \left(e^{-d^2(x,dcc)} \right) \tag{25}$$

Depending on the firing strength of per class (one rule from a class) rules, the class label of x is set by the "winner takes all" formula [23]

$$\text{class label} = \arg \max_{c=1,2....C} (\gamma^c(x)) \tag{26}$$

3.5 Algorithm complexity analysis

The Computational complexity of the offline and online training procedures and the memory space and lifetime of an irrelevant data-cloud in the buffer of the BAFC are analyzed in this subsection.

3.5.1 Offline training process and its computational complexity

According to Algorithm 1, an offline procedure (Section 3.2) is performed on a static dataset, where the computational complexity of the algorithm highly depends on a uniquely identified data sample from the dataset. The computational complexity of the entire offline training procedure is either $\mathcal{O}(\text{c} \times (\text{n} + (\text{n} + (\text{n-1})) + \text{m})$) or $\mathcal{O}(\text{c} \times \text{n})$ (where, always $\text{n} \geq \text{m}$), where c is the number of classes, n is the number of uniquely identified data samples from the static dataset, and m is the number of data-clouds after merging several data-clouds offline.

The ranking process takes at most $\mathcal{O}(\text{n} + (\text{n-1}))$, and offline data-cloud merging takes $\mathcal{O}(\text{m})$. By analyzing the offline training algorithm, one can understand that the data-cloud creation procedure starts with creating the ranking system of a uniquely identified data sample followed by ensuring this structure with its neighboring or supporting data sample using (12).

3.5.2 Online training process and its computational complexity

To compute the complexity of the online training procedure (Section 3.3), the arrived data sample x^k is assumed to belong to class c. Although age calculation (19), buffering (20) and (21) and merging data-clouds (23) increase the computational complexity, the traits ("one pass" type and merging (23) with deletion (22) of data-clouds) of the proposed algorithm minimize such complexity as well. The computational complexity of the online training procedure for the arrived data x^k of class c is $\mathcal{O}(\text{n} + \text{n} + \text{m} + \text{m} + \text{n})$ or $\mathcal{O}(\text{n})$, where n is the number of data-clouds at the k^{th} instance, and m is the number of data-clouds stored in temporary memory (buffer). As the computational complexity of the online training procedure depends on the number of seemingly irrelevant data-clouds, the lifetime of irrelevant data-clouds in the buffer is analyzed in Section 3.5.4. Therefore, the overall complexity of the online training procedure is $\mathcal{O}(\text{n} \times \text{m})$, where n is the total number of arrived data samples as stream data, and m is the number of active data-clouds at the k^{th} time instance.

3.5.3 Memory complexity analysis

The proposed BAFC has memory requirements for storing data-clouds and decay parameters, and for processing data. If an n data-cloud with d dimensional dcc and m number of meta-parameters exits, then BAFC needs a memory of $\mathcal{O}(n \times (m + d) + d)$. Moreover, the memory constraints for meta-parameters and parameter decay are constant, the memory space complexity is $\mathcal{O}(n \times d)$. Figure 6 in Section 4.2.3 shows the memory efficiency of BAFC compared with that of other classification algorithms.

3.5.4 Buffer and lifetime of an irrelevant data-cloud

According to the computational complexity of the online training stage and the analysis of memory complexity, it can appear in mind that an irrelevant data-cloud in the buffer memory always increases the complexity of BAFC. To determine how much time an irrelevant data-cloud stays in the buffer memory, a decay variable is used, which denotes the amount of stream data arriving at a unit time. To retrieve a data-cloud, its age is used along with condition 5 (21) is used to indicate its minimum lifetime in the buffer. As, s_age is 0.5, the maximum life time of a data-cloud in the buffer is $\frac{Decay}{2}$. In addition, Fig. 7 of Section 4.2.4 discusses how the decay variable affects the training time of the classifier.

4 Evaluations

This section discusses the results of an experiment conducted on five real-world datasets, one synthetic dataset, and a well-known image dataset to prove the effectiveness of the proposed BAFC algorithm. The performance of this algorithm is then compared with a number of "state-of-the-art" classifiers using MATLAB 2014a.

4.1 Dataset

To describe the efficiency of BAFC, a well-known benchmark dataset was used, and the performance of BAFC was compared with that of well-established classifiers, including eClass [23]), SVM [48], KNN [5], DT [7],

self-organising map (SOM) [49], DENFIS [50], SOF [4], and TEDAClass [22]. A large set of data helps underscore the capabilities of classifiers on time series, character recognition, and synthetic data classification. The experiment was conducted on the following well-established benchmark and numerical datasets:

- **Occupancy detection**: This dataset [51] contains 20560 instances and 7 attributes for occupied and non-occupied classes.
- **Optical recognition**: This handwritten digit dataset [45] contains 64 features of 5620 instances for 10 digits.
- **Multiple feature**: This dataset [45] consists of 649 features of handwritten numerals (0-9) of 2000 patterns.
- **Letter recognition**: This English capital letter recognition dataset contains 20000 instances with 16 features and 26 class labels [45].
- **SEA**: This synthetic dataset [52], which has the characteristics of concept drift, consists of 50000 instances with 3 features.
- **Electricity pricing**: The electricity market dataset [53] contains 45312 instances and 7 features.
- **Mnist data**: This dataset includes 60000 gray-level training images and 10000 gray-level images for the validation of 10 handwritten digits (0–9). Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels [54].

4.2 Result

BAFC was evaluated against nine well-known classifiers. First, the performance of the BAFC was tested on the six aforementioned benchmark datasets and compared with that of the eight evolving, autonomous, and offline methods. All tabulated outcomes were obtained after 10 runs on a randomized dataset, and the validation stage was used to classify the test dataset. At this stage, the class label of the test dataset was determined using (25) and (26) described in Section 3.4, similar to [4]. The performance of BAFC was evaluated in terms of its i) deletion and merging of data-clouds, ii) training time, iii) memory efficiency, iv) parameter sensitivity, v) accuracy, vi) efficiency on dimensionality, and vii) fineness of data-clouds. The measured performance of BAFC was then compared with that of well-established stream data classifiers, such as AutoClass1 [55], eClass1 [23], TEDAClass [22], SOF [4], and SOM [49] which use data-clouds or AnYa type fuzzy rules to define the fuzzy rule base. These classifiers were selected because they are state-of-the-art models for classifying stream data in the field of data classification using the FRB system.

4.2.1 Deletion and merging of data-clouds

As the occupancy detection dataset and optical recognition of handwritten digits dataset contains both training and testing sets, 15% of the training data were used for the offline training, and the rest was used for the online training. For the multiple features, SEA, letter recognition, and electricity prizing datasets 15% of the overall data were used for offline training, 50% were used for online training, and 35% of data were used for validation.

Figure 3, presents the situation before and after the online training in the data space. To illustrate the online pruning process, offline and online data-cloud merge operation, 1000 data sample of SEA data (class 1) were used. Another 1000 pieces of data were used as stream data to highlight the operations, where the decay value is 500. Among huge number of irrelevant data-cloud, a few is highlighted here (Fig. 3c).

The minimization and deletion of a huge amount of data-clouds which are the focus points of BAFC, were tested on the occupancy detection dataset. Figure 4 shows the number of active data-clouds, the data-clouds in the buffer, and the deleted or dead data-clouds during the online training stage of the occupancy detection dataset for class 1. The offline training stage produced more or less than 150 data-clouds at the online training stage and, after reading 2000 data samples online, the number of data-clouds increased as new data samples were used as stream data as more data-clouds were merged or removed from the buffer. The online training process eventually produced more or less than 130 data-clouds for the validation stage. Equations (14), (22), and (23) plays the most important roles in reducing or deleting irrelevant and overlapping data-clouds.

4.2.2 Training time

The total training time was measured based on the offline and online training times. Figure 5 shows the average training time for all large datasets, namely, MNIST, electricity pricing, occupancy detection and SEA datasets. Figure 5a compares the training time of a single data from the MNIST dataset with that of two prototype-based and two data-cloud-based evolving algorithms, namely, AutoClass1 [55], eClass1 [23], TEDAClass [22], and SOF [4]. As shown in Fig. 5a, the training times of TEDAClass, eClass1, and Autoclass1 were considerably high than those of BAFC and SOF, whereas the initial training time of BAFC was longer than those of SOF and eClass1. In Figs. 5b, 5c and 5d compares the training times per data of the electricity pricing, occupancy detection, and SEA datasets with those of eClass1, TEDAClass, SOF, and SOM [49], respectively.

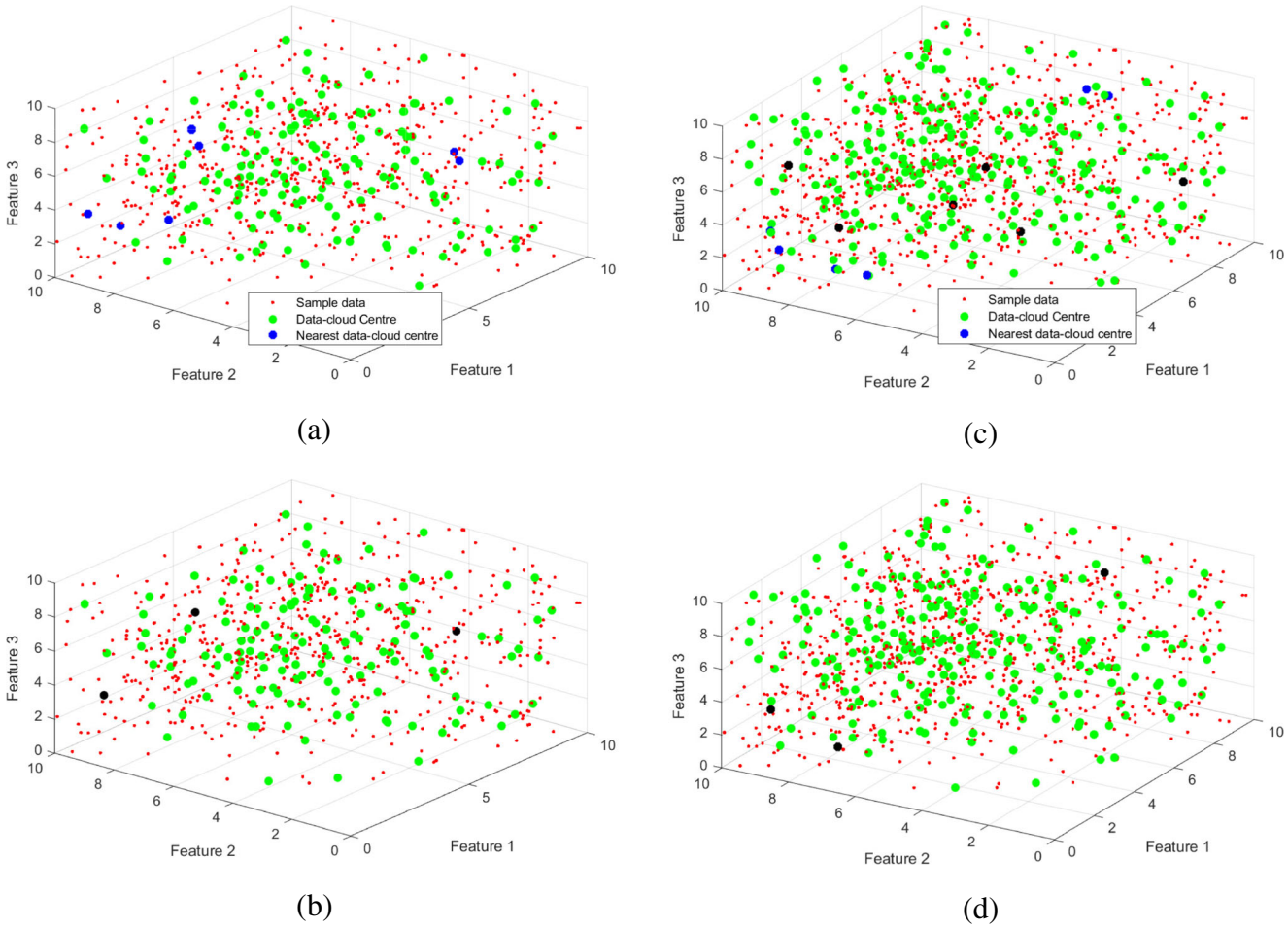


Fig. 3 Data-clouds (green asterisks) and data samples (red dots). (a) The blue asterisks are the nearest data-clouds that will be merged according to condition-2, (b) the black asterisks are the merged data-clouds at the offline training stage, (c) the black asterisks are the irrelevant data-clouds in the buffer while stream data are being fetched online, and the blue asterisks are the nearest data-clouds that will be merged according to condition-7. (d) After fetching additional stream data, the irrelevant data-clouds are pruned according to condition-6, and the blue asterisks from Fig. 3c are merged into black asterisks according to condition-7

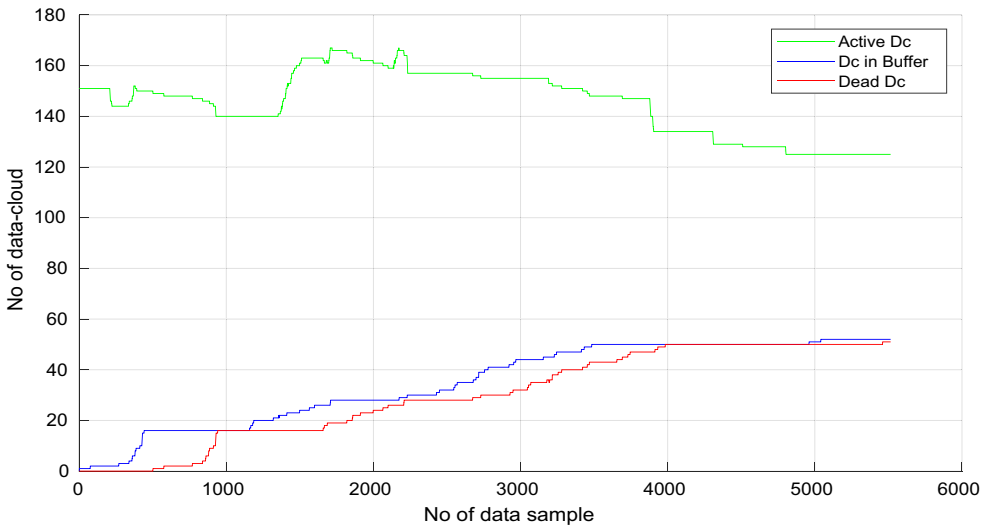


Fig. 4 Reading of data-cloud number during the evolving training phase for class 0

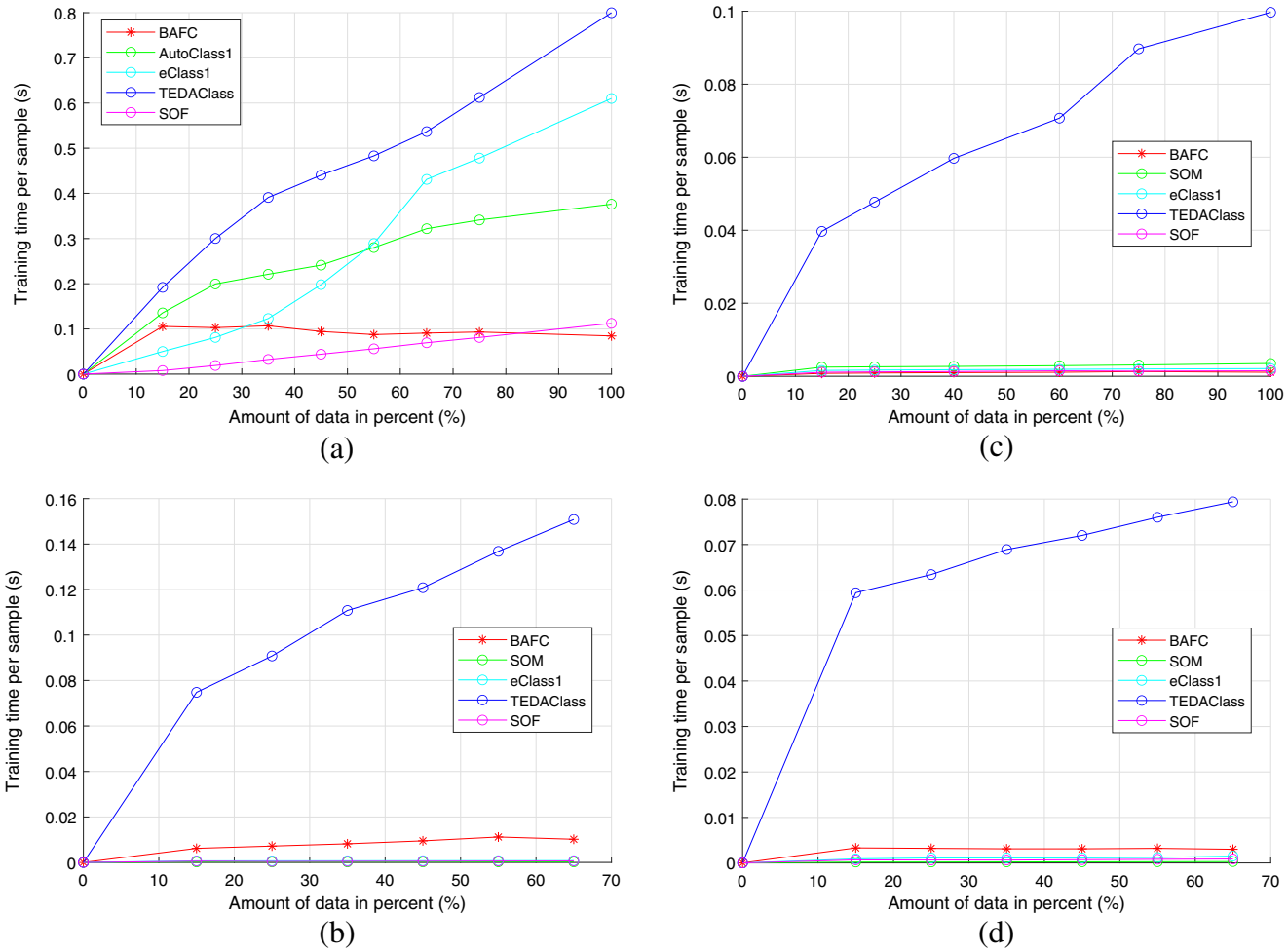


Fig. 5 Per sample training time graph for the (a) MNIST, (b) electricity pricing, (c) occupancy detection and (d) SEA datasets

For the electricity pricing (Fig. 5b) and SEA datasets (Fig. 5d), BAFC outperformed TEDAClass whereas for the occupancy detection dataset (Fig. 5c), BAFC outperformed all other classifiers. Although a large amount of time is required for the online data-cloud deletion (22) and merging operations (23), BAFC recovers (20) some relevant data-clouds from the buffer instead of creating new data-clouds by minimizing the training time. The algorithm also reduces the number of data-clouds by performing merging operations offline (condition 2 (14)) and online (condition 7 (23)), hence further reducing the training time. The computational complexity and number of prototypes or datapoints in BAFC explain this result.

4.2.3 Memory efficiency

Section 3.5.3 shows that the memory requirements of a data-cloud-based classification algorithm are directly related to the number of data-clouds. The number of data-clouds in the proposed BAFC was then compared with that in SOF and TEDAClass when training the MNIST, electricity pricing, occupancy detection, and SEA datasets online to demonstrate the memory efficiency of the proposed algorithm. The memory efficiency of BAFC for the abovementioned datasets is illustrated in Fig. 6a, b, c and d, respectively.

As shown in Fig. 6, the number of created data-clouds in SOF increased with time because this classifier does not have any online pruning and data-cloud merging systems for irrelevant and overlapping data-clouds. Similar to SOF, TEDAClass does not have an online pruning system for irrelevant data-clouds. As a result, both SOF and TEDAClass show a comparatively poor memory efficiency. Meanwhile, BAFC achieves memory efficiency by merging overlapping data-clouds in both offline and online (conditions-2 and 7) and removing irrelevant data-clouds while processing stream data online (condition-6).

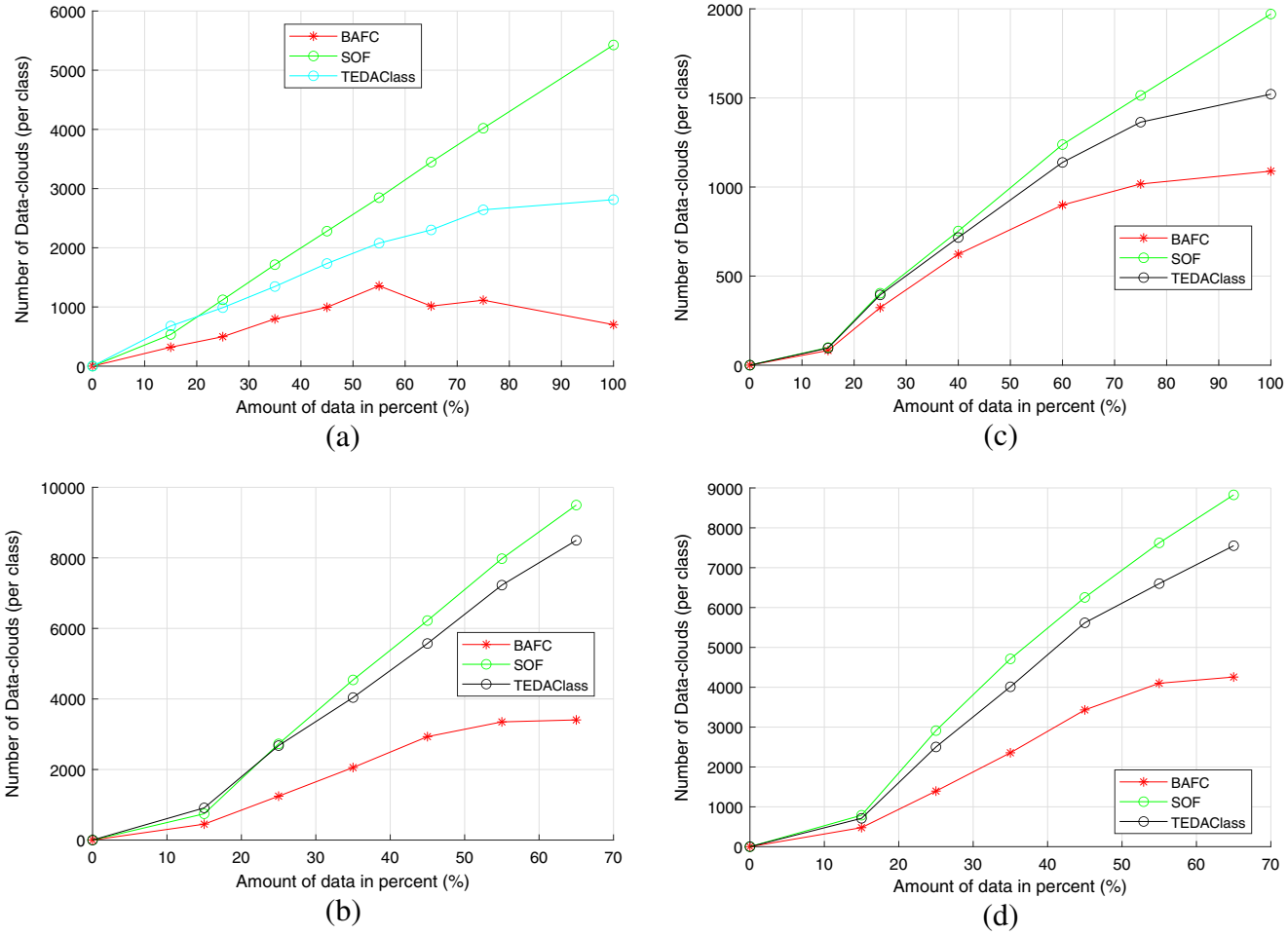


Fig. 6 Memory use during the training of the (a) MNIST, (b) electricity pricing, (c) occupancy detection, and (d) SEA datasets

BAFC does not create data-clouds frequently but instead retrieves data-clouds from the buffer (condition-5). Figure 6 highlights the superiority of BAFC over SOF and TEDAClass. BAFC also uses less than half of the memory consumed by TEDAClass and SOF for storing data-clouds.

4.2.4 Parameter sensitivity

The total training time is recorded for various decay values of MNIST data to assess the parameter decay behavior of BAFC. Figure 7 shows the performance of BAFC for

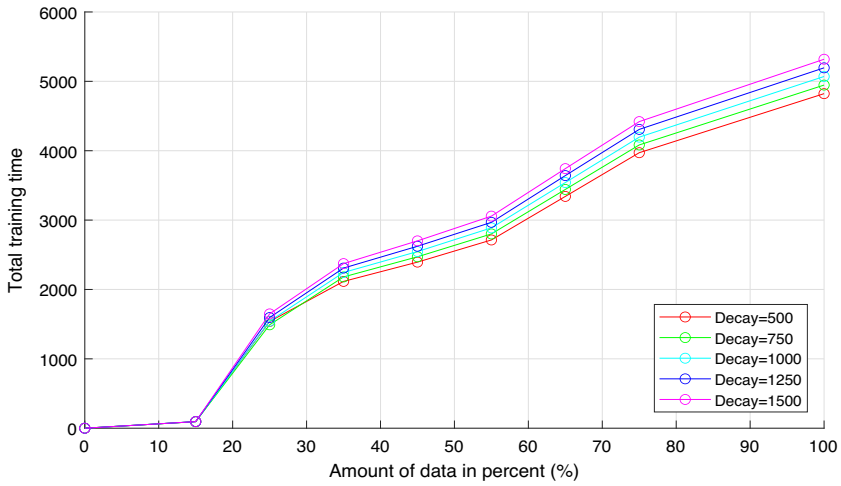


Fig. 7 Total training time for various decay parameter values

different decay values. The reduction in s_age at the online training stage (Algorithm 2, Section 3.3) indicates that with a larger value of decay, data-clouds stored in the buffer waits for a long time. Therefore, training time and decay are proportional to each other. According to Section 3.5.2, extra processing time increases for a larger value of decay. Figure 7 also shows the relationship between decay and training time at the online training stage, in which the total training time decreases when the BAFC uses smaller decay values.

4.2.5 Accuracy

While comparing BAFC with state-of-the-art classifiers, a cosine dissimilarity with the level of granularity, G = 12 was used for SOF, a linear kernel was used for SVM classifier, k = 10 was used for the KNN, and the "winner-takes-all" concept with a net size of 9 × 9 was applied for SOM.

Along with the data-clouds created at the online training stage, (25) and (26) (Sections 3.2, 3.3, and 3.4) are used to determine the class label of the test dataset. After determining the class label, the classification accuracy is measured based on the total number of correct predictions and amount of data in the test dataset.

In Table 1, Acc denotes the accuracy achieved by a classifier in a dataset, and $Time(s)$ denotes the required for training a data sample in seconds. The evaluation results clearly reveal that BAFC outperformed all other classifiers, with accuracies of 94.96% and 77.08% for the letter recognition and electricity pricing datasets, respectively. For the occupancy detection dataset, KNN achieved a remarkable performance with accuracy of 96.64%, whereas for the optical recognition dataset, SOF obtained the highest accuracy of 98.39%. Meanwhile, the offline SVM algorithm outperformed all other classifiers in both the multiple feature and SEA datasets with accuracies of 96.71% and 82.65%, respectively, by training 65% of the data and using the remaining 35% for testing. KNN, SVM, and DT are offline classifiers that use only static datasets and are unable to classify stream data. In this experiment, eClass obtained the lowest accuracy among all classifiers, whereas DENFIS failed in the classification task due to the high dimensionality of the optical recognition and multiple feature datasets [4]. In sum, BAFC can compete with state-of-the-art classifiers with an average accuracy of 90.82%.

Table 2 presents the classification rate used to evaluate the accuracy of the six classifiers for the MNIST [54] dataset. Various sizes (5% to 100%) of training sets were used online and offline. To conduct the online training, 10% of the data were trained offline and produced a comparable accuracy of 97.13%. When using image data, BAFC achieved a significantly better performance compared with the other classifiers even for the 15% data used at the online training stage. BAFC and SVM also obtained more stable results with SOF, eClass1, Autoclass1, and TEDAclass. In case of SVM, all data were trained as a static dataset. The performance of BAFC on an image dataset can be further improved by using different types of image feature descriptors, such as LNDP [56].

4.2.6 Efficiency on dimensionality

Proving that BAFC is free from the curse of dimensionality is outside the scope of this research theoretically. However, this work attempts to prove this point experimentally by analyzing the effect of dissimilarity measurements on three benchmark datasets. Both Euclidean and cosine dissimilarity measurements were used for BAFC, and the results are plotted in Fig. 8. Two datasets with large features, namely, MNIST and Multiple Features , were used along with a high dimensional dataset, UCMerced [57], which is another version of the multiple features dataset with a higher dimension, were used to prove the efficiency of BAFC. The remote sensing image dataset UCMerced contains 21 classes, with each class having 100 images of 256 x 256 pixel. For this dataset, 15% of the data were used at the offline training stage, 5% to 50 % of data were used at the online training stage, and the rest of the data were used at the validation stage.

Following the common practice, for the UCMerced dataset, a high level ensemble was created by a pre-trained AlexNet [58] and the VGG-VD-16 [59] deep neural network for image feature extraction. The ensemble descriptor extracted a 1 x 9192 highly discriminative representation vector from image I as

$$F(I) = \left[\frac{AN(I)}{\mid AN(I) \mid}, \frac{VV(I)}{\mid VV(I) \mid}\right]^{\mathrm{T}} \tag{27}$$

where AN(I) and VV(I) denote the 1 x 4096 dimensional activations derived from the first fully connected layer of the two DNN models. These extracted features were then used to train and validate BAFC.

Figure 8 shows the accuracy of BAFC with different amounts of data for the MNIST, multiple features, and UCMerced datasets as measured by cosine and Euclidean dissimilarity. Cosine dissimilarity was proven to be more effective than Euclidean distance in the case of high-dimensional problems. The use of cosine dissimilarity in the recursive formula for calculating unimodal density also enhanced the efficiency of BAFC.

4.2.7 Fineness of data-clouds

Another advantage of BAFC is its acquisition of extremely transparent, human-understandable AnYa type fuzzy rules due to its data-cloud-based structure. Table 3 presents the

Table 1 Accuracy of different classifiers on each dataset

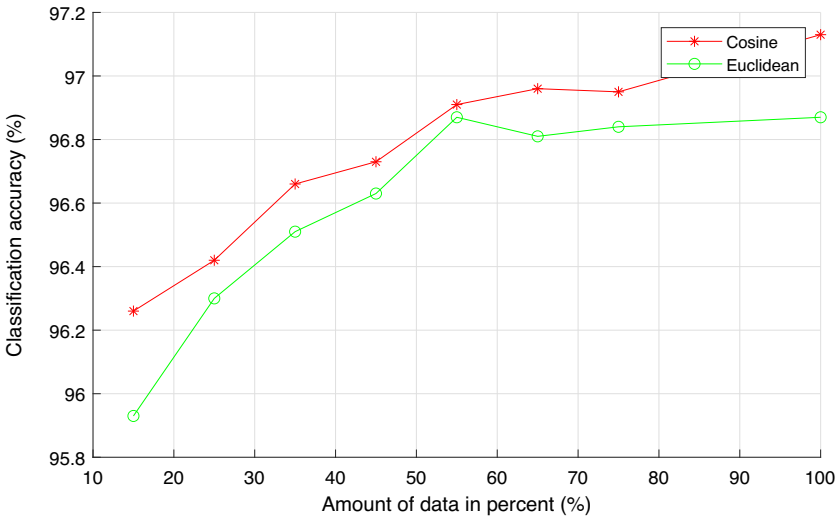
Algorithm / Data		Letter recognition	Occupancy detection	Optical recognition	Multiple features	Electricity pricing	SEA	Average, %
Proposed	Acc	**94.96**	96.03	97.90	96.61	**77.08**	82.38	**90.82**
	Time(s)	4.09	9.32	4.81	13.49	301.39	113.81	
SOF [4]	Acc	92.98	95.88	**98.39**	93.66	71.39	74.56	87.81
	Time(s)	2.19	9.70	4.19	2.65	19.91	28.31	
eClass [23]	Acc	51.25	88.63	86.81	82.64	67.27	72.34	74.82
	Time(s)	9.76	17.72	15.61	6.59	24.10	45.12	
KNN [5]	Acc	91.80	**96.64**	97.66	91.51	72.20	73.03	87.14
	Time(s)	3.05	4.91	4.08	3.02	5.05	15.09	
TEDA-Class [22]	Acc	51.54	96.34	91.20	86.37	72.63	76.22	79.05
	Time(s)	3034.32	812.11	2217.19	18001.44	4442.6	2581.41	
SVM [48]	Acc	85.33	95.77	96.27	**96.71**	71.11	**82.65**	87.97
	Time(s)	56.34	311.23	17.94	46.71	66.14	289.23	
DT [7]	Acc	82.43	93.14	85.25	92.44	68.18	69.91	81.89
	Time(s)	2.49	4.11	3.61	2.94	3.79	3.99	
DENFIS [50]	Acc	32.56	89.09	No valid result	No valid result	51.08	63.40	–
	Time(s)	198.81	46.81			231.34	216.51	
SOM [49]	Acc	54.49	94.71	95.03	74.91	64.05	80.56	77.29
	Time(s)	45.81	28.41	31.08	69.92	6.41	9.41	

The best performance in accuracy for a specific dataset for any classification algorithm is emphasized

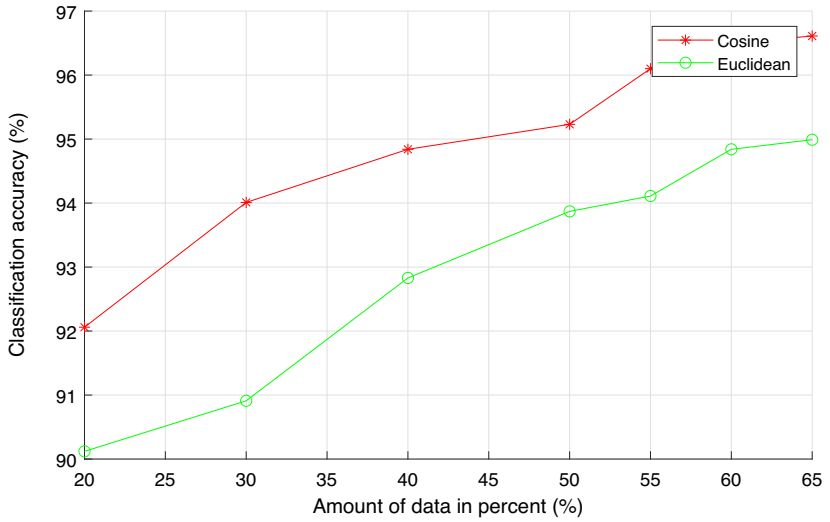
Table 2 Performance comparison (in accuracy, %) with different amounts of data samples in the MNIST dataset

Amount of training sample	BAFC online	BAFC offline	SOF [4]	SVM [48]	Autoclass1 [55]	eClass1 [23]	TEDA-Class [22]
5%		**96.11**	95.66	96.03	94.51	94.37	94.63
10%		96.19	95.87	**96.21**	94.81	94.55	95.01
15%	96.26	96.21	95.99	**96.29**	95.02	94.83	95.12
25%	**96.42**	96.29	96.12	96.41	95.13	94.91	95.26
35%	**96.66**	96.54	96.18	96.59	95.31	95.04	95.31
45%	**96.73**	96.66	96.23	96.63	95.39	95.31	95.33
55%	**96.91**	96.78	96.21	96.88	95.35	95.39	95.51
65%	**96.96**	96.89	96.39	96.95	95.38	95.41	95.63
75%	96.95	96.81	96.54	**97.01**	95.39	95.42	95.67
100%	97.13	97.01	96.70	**97.15**	95.44	95.41	95.61

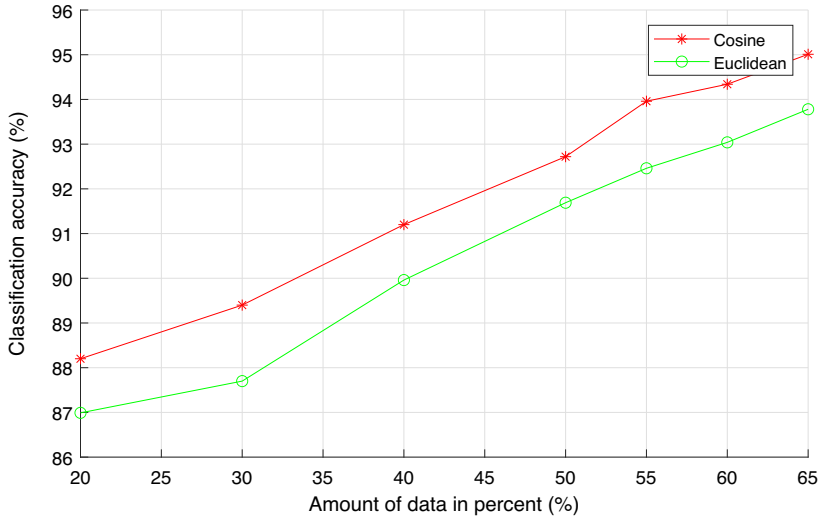
The best performance in accuracy for different amounts of MNIST data is identified by making it bold



(a)



(b)



(c)

Fig. 8 Accuracy based on two dissimilarity measurement processes for the (a) MNIST, (b) multiple features, and (c) UCMerced datasets

Table 3 A Part of the fuzzy-rule generated from the evolving stage

Dataset	IF part of AnYa type Fuzzy rule	THEN part
SEA	x ∼ (8.231264, 6.158971, 0.652521) **OR** x ∼ (3.240307, 9.291315, 0.819171) **OR .... OR** x ∼ (8.131284, 5.072553, 3.631046)	Class 1
	x ∼ (0.263657, 1.5065, 7.252238) **OR** x ∼(0.356543, 0.585792, 8.048625) **OR .... OR** x ∼ (1.659741, 5.539075, 0.534608)	Class 2
Electricity Pricing	x ∼ (0.148936, 0.0455798, 0.149194, 0.003467, 0.422915, 0.414912) **OR** x ∼ (0.127660, 0.045717, 0.151032, 0.003467, 0.422915, 0.414912) **OR ..**	Class 1
	.. OR x ∼ (0.042553, 0.053233, 0.242189, 0.003467, 0.422915, 0.414912) **OR** x ∼ (0.170213, 0.046265, 0.155432, 0.003467, 0.422915, 0.414912)	
	x ∼ (0.423292, 0.090428, 0.532360, 0.003467, 0.422915, 0.414912) **OR** x ∼ (0.189249, 0.072599, 0.209023, 0.003467, 0.422915, 0.414912) **OR ..**	Class 2
	.. OR x ∼ (0.382979, 0.084680, 0.539956, 0.003467, 0.422915, 0.414912) **OR** x ∼ (0.240300, 0.087543, 0.366233, 0.003467, 0.422915, 0.414912)	
MNIST	x ∼  **OR** x ∼ **OR** x ∼ **OR .... OR** x ∼ **OR** x ∼	Class 1
	x ∼  **OR** x ∼ **OR** x ∼ **OR .... OR** x ∼ **OR** x ∼	Class 4
	x ∼  **OR** x ∼ **OR** x ∼ **OR .... OR** x ∼ **OR** x ∼	Class 7
	x ∼  **OR** x ∼ **OR** x ∼ **OR .... OR** x ∼ **OR** x ∼	Class 10

The bold entries are used to understand the fuzzy rule base

fuzzy rules generated from the SEA, electricity pricing, and MNIST datasets in the online process of BAFC. The data-cloud center of the SEA dataset was created using its three features, and the data-cloud center of the electricity pricing dataset was created using its six features. For the MNIST dataset each presented image was treated as a data-cloud center, and created by its 784-pixel value after the online training process. For clarity, the images extracted from the AnYa type fuzzy rule in the table were scaled. The scaled image of the rules extracted from the MNIST dataset expresses the great compactness and transparency of BAFC.

5 Conclusions

This study proposes BAFC based on data-clouds and the AnYa type FRB system. BAFC does not need prior assumptions about the dataset and has a highly transparent structure composed of meaningful data-clouds. This classifier can recognize potential data-clouds from the data sample used in offline training in an effective and efficient way and continue learning from the data by using newly introduced recursive formulas for calculating density. BAFC also consumes comparatively more memory

than the offline- and online-prototype-based SOF by storing temporary irrelevant data-clouds in the buffer. However, the proposed algorithm obtains considerably better results than any of the other classifiers and can distinguish the changes in the arriving stream data by utilizing the online data-cloud merging concept. The evolving online training structure of BAFC can change the properties of data-clouds and even create new data-clouds to adopt the changes in stream data. Experimental results from the benchmark dataset highlight the excellent performance of BAFC and its ability to handle a variety of situations. BAFC therefore offers a suitable alternative to conventional algorithms given its excellent accuracy, transparency, fast-evolving learning techniques, and applicability to stream numerical and image data.

In the future, the BAFC will be extended to an AnYa type FRB system where data-clouds can be used in a hierarchical order. The hierarchy of data-clouds will help efficiently identify overlapping data-clouds and particular classes. Experiments for this system can be conducted on medical, intrusion detection, and image datasets. Another relevant process for a particular dataset can be applied where the average distance of a data-cloud will be fixed with lower and higher ranges of values to classify data.

Acknowledgements The authors would like to acknowledge the Qatar National Library for the Open Access funding. This work was also supported by Special Grant of ICT Division (Ministry of Posts, Telecommunications and Information Technology), Bangladesh, and Grant No. 56.00.0000.028.20.004.20-333.

References

1. Lughofer E (2021) Improving the robustness of recursive consequent parameters learning in evolving neuro-fuzzy systems. In: Information sciences. vol 545, pp 555–574
2. Ge D, Zeng X-J (2020) Learning data streams online–An evolving fuzzy sys- tem approach with self-learning/adaptive thresholds. In: Information sciences. vol 507, pp 172–184
3. Hariri RH, Fredericks EM, Bowers KM (2019) Uncertainty in big data analytics: survey, opportunities, and challenges. In: Journal of big data 6.1, pp 44
4. Gu X, Angelov PP (2018) Self-organising fuzzy logic classifier. In: Information Sciences. vol 447, pp 36–51
5. Cunningham P, Delany SJ (2021) k-Nearest neighbour classifiers-A tutorial. In: ACM computing surveys (CSUR) 54.6, pp 1–25
6. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer science & business media
7. Safavian SR, Landgrebe D (1991) A survey of decision tree classifiser methodology. In: IEEE transactions on systems, man, and cybernetics 21.3, pp 660–674
8. Subudhi A, Dash M, Sabut S (2020) Automated segmentation and classification of brain stroke using expectation-maximization and random forest classifier. In: Biocybernetics and biomedical engineering 40.1, pp 277–289
9. Wang M, Chen H (2020) Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis. In: Applied soft computing. vol 88, p 105946
10. Gibert D et al (2019) Using convolutional neural networks for classification of malware represented as images. In: Journal of computer virology and hacking techniques 15.1, pp 15–28
11. Yadav SS, Jadhav SM (2019) Deep convolutional neural network based medical image classification for disease diagnosis. In: Journal of big data 6.1, pp 1–18
12. Alimjan G et al (2018) A new technique for remote sensing image classification based on combinatorial algorithm of SVM and KNN. In: International journal of pattern recognition and artificial intelligence 32.07, p 1859012
13. Cervantes J et al (2020) A comprehensive survey on support vector machine classification: applications, challenges and trends. In: Neurocomputing, vol 408, pp 189–215
14. Fahim SR et al (2020) Microgrid fault detection and classification: machine learning based approach, comparison, and reviews. In: Energies 13.13, pp 3460
15. Dash S et al (2019) A Neuro-fuzzy approach for user behaviour classification and prediction. In: Journal of cloud computing 8.1, pp 1–15
16. Assegie TA, Nair PS (2019) Handwritten digits recognition with decision tree classification: a machine learning approach. In: International journal of electrical and computer engineering (IJECE) 9.5, pp 4446–4451
17. Ali W et al (2021) Classical and modern face recognition approaches: a complete review. In: Multimedia tools and applications 80.3, pp 4825–4880
18. Yeganejou Mojtaba, Dick Scott, Miller James (2019) Interpretable deep convolutional fuzzy classifier. In: IEEE transactions on fuzzy systems 28.7, pp 1407–1419
19. Lughofer E, Pratama M, Škrjanc I (2021) Online bagging of evolving fuzzy systems. In: Information sciences 570, pp 16–33
20. Muthugala MA et al (2020) A self-organizing fuzzy logic classifier for benchmarking robot-aided blasting of ship hulls. In: Sensors 20.11, p 3215
21. Tsakiridis NL et al (2019) An evolutionary fuzzy rule-based system applied to the prediction of soil organic carbon from soil spectral libraries. In: Applied soft computing 81, pp 105504
22. Kangin Dmitry, Angelov Plamen, Iglesias JA (2016) Autonomously evolving classifier TEDAClass. In: Information sciences 366, pp 1–11
23. Angelov PP, Zhou Xiaowei (2008) Evolving fuzzy-rule-based classifiers from data streams. In: Ieee transactions on fuzzy systems 16.6, pp 1462–1475
24. Angelov Plamen P, Gu Xiaowei, Principe JC (2017) Autonomous learning multimodel systems from data streams. In: IEEE transactions on fuzzy systems 26.4, pp 2213–2224
25. Noorbehbahani Fakhroddin et al (2017) An incremental intrusion detection system using a new semi-supervised stream classification method. In: International journal of communication systems 30.4, pp e3002
26. Iglesias JA et al (2011) Creating evolving user behavior profiles automatically. In: IEEE transactions on knowledge and data engineering 24.5, pp 854–867
27. Angelov Plamen, Yager Ronald (2012) A new type of simplified fuzzy rule-based system. In: International journal of general systems 41.2, pp 163–185
28. Senoussaoui M et al (2013) Efficient iterative mean shift based cosine dissimilarity for multi-recording speaker clustering. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 7712–7715
29. Aggarwal CC, Hinneburg Alexander, Keim Daniel (2001) A On the surprising behavior of distance metrics in high dimensional space. In: International conference on database theory. Springer, pp 420–434

30. Prim RC (1957) Shortest connection networks and some generalizations. In: The bell system technical journal 36.6, pp 1389–1401
31. Angelov P et al (2016) Empirical data analysis: a new tool for data analytics. In: 2016 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, pp 000052–000059
32. Angelov P, Zhou X, Klawonn Frank (2007) Evolving fuzzy rule-based classifiers. In: 2007 IEEE symposium on computational intelligence in image and signal processing. IEEE, pp. 220–225
33. Shahparast Homeira, Mansoori EG, Jahromi MZ (2019) AFCGD: an adaptive fuzzy classifier based on gradient descent. In: Soft computing 23.12, pp 4557–4571
34. Ahmed Usman, Lin* JC-W, Srivastava Gautam (2022) Fuzzy contrast set based deep attention network for lexical analysis and mental health treatment. In: Transactions on asian and low-resource language information processing
35. Wu M-E et al (2021) Effective fuzzy system for qualifying the characteristics of stocks by random tradings. In: IEEE transactions on fuzzy systems
36. Wu T-Y et al (2020) An efficient algorithm for fuzzy frequent itemset mining. In: Journal of intelligent & fuzzy systems 38.5, pp 5787–5797
37. Qasem SN et al (2021) A type-3 logic fuzzy system: Optimized by a correntropy based Kalman filter with adaptive fuzzy kernel size. In: Information sciences. vol 572, pp 424–443
38. Wang J-h et al (2021) Non-singleton type-3 fuzzy approach for flowmeter fault detection: experimental study in a gas industry. In: Sensors 21.21, pp 7419
39. Anter AM et al (2020) A new type of fuzzy-rule-based system with chaotic swarm intelligence for multiclassification of pain perception from fMRI. In: IEEE transactions on fuzzy systems 28.6, pp. 1096–1109
40. Angelov P, Gu X, Kangin D (2017) Empirical data analytics. In: International journal of intelligent systems 32.12, pp 1261–1284
41. Gu X, Angelov PP, Principe JC (2018) A method for autonomous data par- titioning. In: Information sciences. vol 460, pp 65–82
42. de Campos Souza PV, Wang Y-K, Lughofer E (2020) Knowledge extraction about patients surviving breast cancer treatment through an autonomous fuzzy neural net- work. In: 2020 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE, pp 1–8
43. Calabrese F et al (2020) Unsupervised fault detection and prediction of remaining useful life for online prognostic health management of mechanical systems. In: Applied sciences 10.12, pp 4120
44. Angelov PP, Gu X (2018) Empirical fuzzy sets. In: International journal of intelligent systems 33.2, pp 362–395
45. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 3 March 2022
46. Boots B, Okabe A, Sugihara K (1999) Spatial tessellations. In: Geographical information systems vol 1, pp 503–526
47. Islam MK, Ahmed MM, Zamli KZ (2019) A buffer-based online clustering for evolving data stream. In: Information sciences, vol 489, pp 113–135
48. Cristianini Nello, Shawe-Taylor John et al (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press
49. Płoński P, Zaremba K (2012) Self-organising maps for classification with metropolis-hastings algorithm for supervision. In: International conference on neural information processing. Springer, pp 149–156
50. Kasabov NK, Song Q (2002) DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. In: IEEE transactions on fuzzy systems 10.2, pp 144–154
51. Candanedo LM, Feldheim V (2016) Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. In: Energy and buildings 112, pp. 28–39
52. Street WN, Kim Y (2001) A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp 377–382
53. Holmes RKABG, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11:1601–1604. http://portal.acm.org/citation.cfm?id=1859903. Accessed 3 March 2022
54. Cortes C, LeCun Y MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/,in:2010. Accessed 3 March 2022
55. Angelov P et al (2014) Symbol recognition with a new autonomously evolving classifier au- toclass. In: 2014 IEEE conference on evolving and adaptive intelligent systems (EAIS). IEEE, pp 1–7
56. Verma M, Raman B (2018) Local neighborhood difference pattern: a new feature descriptor for natural and texture image retrieval. In: Multimedia tools and applications 77.10, pp 11843–11866
57. Yang Y, Newsam S (2010) Bag-of-visual-words and spatial extensions for land-use classification. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems, pp 270–279
58. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol 25
59. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations, pp 1–14

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Affiliations

Sajal Debnath[1] · Md Manjur Ahmed[1] · Samir brahim Belhaouari[2] · Toshiyuki Amagasa[3] · Mostafijur Rahman[4]

✉ Samir brahim Belhaouari
sbelhaouari@hbku.edu.qa

Sajal Debnath
sajal.cse3.bu@gmail.com

Toshiyuki Amagasa
amagasa@cs.tsukuba.ac.jp

Mostafijur Rahman
mostafijur@cse.green.edu.bd

1 Department of Computer Science and Engineering, University of Barishal, Barishal-8254, Barishal, Bangladesh

2 Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar

3 Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan

4 Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh